Are Distributed Peer-to-Peer Overlay Networks Worth The Effort?

Jacob Thebault-Spieker
Computer Science Department
University of Minnesota, Morris
600 E. 4th St.
Morris, MN 56267
theba004@morris.umn.edu

ABSTRACT

Distributed peer-to-peer overlay networks are not dependent on any one point of failure, can be "overlayed" on top of a traditional network (allowing a subset of machines to form their own network within the larger network), and do not require their own specific infrastructure. There are a number of distinct types of distributed peer-to-peer overlay networks, including structured, unstructured, and hybrid overlay networks. We will discuss the similarities and differences between these types of overlay networks, as well as discuss the benefits and downsides to overlay networks as a whole. Further, we will show how distributed peer-to-peer overlay networks can be applied, given the benefits they provide.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

General Terms

overlay networks, DHT, peer-to-peer

Keywords

Peer-to-peer, overlay networks, distributed systems, structured, unstructured, overlay

1. INTRODUCTION

Distributed peer-to-peer (dP2P) overlay networks can be used for many different things, but ultimately they provide a way of interacting with certain machines on the Internet, while excluding others. Often times, these networks are specifically intended to be exclusive, although this is not a necessary factor. These networks are most commonly used for file sharing, although can be used for anything a normal networked computer would be used for.

The basic use for dP2P overlay networks is a simple, distributed key-value store (that is, a peer looks up a key, and receives the value attached to the key). In this usage case, the value may be a simple message to pass to a destination node, or it may return the IP address and file system path for a requested file. In a simple message passing context, the message can be considered equivalent to a standard packet in traditional networking. Therefore, the dP2P overlay network may be used for anything that a traditional network would be used for. Commonly, dP2P systems are used for filesharing networks (as we have indicated elsewhere), but may be used for wide-area distributed file stores like PAST [1], or OceanStore [3].

However, because of the generic nature of key-value lookup, file storage and sharing is not the only use for dP2P networks. Millar et al.[6] demonstrate a peer-to-peer overlay network to allow quick deployment of a mobile ad-hoc emergency response network. In particular, the algorithm, referred to as Common Group Aliasing (or CGA) is used in conjunction with Bamboo, a structured dP2P overlay network based on Pastry (see Section 6.1 for discussion of Pastry). Millar et al. developed this system in an attempt to deal with extreme emergency cases (the authors specifically reference the EU-FP7 PEACE project as a source for these types of emergency situations). Traditionally, when emergency responders needed use of a network, they would be required to provide their own infrastructure at the site. Overlay networks, in particular dP2P overlay networks are uniquely suited to this type of need, given that they inherently are not dependent on any infrastructure outside of the requirements of the individual peers, and allow more time to be spent on responding to the emergency.

Proper discussion of distributed peer-to-peer overlay networks requires specific distinctions to be made between distributed and traditional networks, peer-to-peer and traditional client-server systems, and overlay networks and physical, individual networks. In order to do this, we will first discuss the differences between a physical network and an overlay network in Section 2. From there, in Section 3 we will define what a distributed peer-to-peer overlay network is and how it differs from other types of overlay networks.

We will then discuss the three primary types of distributed peer-to-peer overlay networks (unstructured, hybrid, and structured) and provide examples of these types of networks. This will provide enough context to be able to discuss specific types of dP2P overlay networks, and some of the benefits and downsides of dP2P overlay networks when compared to more traditional networking architectures. This will be done in Sections 4, 5, and 6 respectively. We will then finish up in Section 7 with an analysis of research being done on distributed peer-to-peer overlay networks, and potential future research areas.

Computer Science Department Senior Seminar Conference '10 Morris, Minnesota USA.

2. BACKGROUND

In order to understand overlay networks, we must first make the distinction between the network and the overlay. The physical network consists of the network as it exists in a physical sense, including things such as: large backbones of fiber optics, routers, servers, and home user connections. The physical network, on a global scale, makes up the Internet, and allows computers from anywhere in the world to interact with one another. Prior to the Internet, computers could be networked together, but those networks were self-contained, and it was impossible for those computers to interact with machines in a separate network (at a different university, for instance). Overlay networks are used as solutions for anything from emergency response systems to applications for smartphones. Ultimately, an overlay network can be used for almost any network requirement, unless the solution needs a physical disconnect between it's network and that of the rest of the world.

2.1 Definition of an Overlay Network

An overlay network can be thought of as the result of applying a filter to the entire Internet. That is, an overlay network is a group of machines connected to the Internet which all meet the criteria of the filter, and form a network with the other Internet-connected machines that also meet the criteria of the filter. This network would be an overlay network, because it is "laid over" the Internet, creating a network which is a subset of the global Internet. An overlay network takes advantage of the fact that the infrastructure needed for the participating machines to interact is already in place. For instance, the group of people who frequent a website at regular intervals (or, more accurately the machines they use to access the website) and the web server(s) running that website form an overlay network. Unfortunately, this definition includes every client-server system (web server and browser, email server and client, etc.). Additionally, defining an overlay network this way means that a subset of Internet-connected machines interacting with subset of machines smaller than the size of the Internet is technically interacting via an overlay network. Users may be participating (by doing something as simple as surfing many websites at the same time) in multiple overlay networks at once. In this paper, we will focus on a specific type of overlay network: peer-to-peer (P2P), distributed overlay networks.

2.2 What are Distributed Peer-to-Peer Overlay Networks?

In distinguishing distributed, peer-to-peer overlay networks from the very broad pool of overlay networks, there are two crucial factors. First, in order for a system to be distributed, it cannot have a single point of failure. In a traditional client-server model (web server and browser, for instance), if the server fails, the client is unable to access the information provided by the server.

Second, within a peer-to-peer system, there is no central node or set of nodes within the network, and each node is given the designation "peer". There are no servers, and there are no clients in a peer-to-peer system, only peers interacting with other peers. A distributed, peer-to-peer overlay network will be referred to hereafter as a dP2P overlay network.

A well known example of a dP2P overlay network is the file-sharing service BitTorrent. [11] For a user to be able to gain access to the BitTorrent network, the user must install BitTorrent specific software on their computer. This software allows the user to join the BitTorrent network, and provids them the means to understand and use the BitTorrent protocol. In order to share files using BitTorrent, one peer must create a torrent file, which provides the information about which files are being shared, as well as information about the "tracker" (a server which coordinates distribution of data between peers). BitTorrent operates across the Internet, and the individual peers within BitTorrent only interact with other BitTorrent peers (within the BitTorrent network). The peers within the BitTorrent network are not limited to participating only within the BitTorrent network, but may also browse the web, check their email, etc. However, within the context of BitTorrent, the machine being used to share files is a single peer within the BitTorrent network, connecting to other peers and sharing data with these peers and only these peers. A person visiting the GMail website will not improperly receive the BitTorrent data unless they are also a part of the BitTorrent network.

When a user (or peer) requests data from the BitTorrent network, the software keeps track of the other peers within the BitTorrent network, and downloads the data it requires from the peers that have the data being requested. BitTorrent downloads sections of the total requested data (referred to as data chunks), potentially from entirely different peers, re-assembles this data, and provides the user with the full requested file upon completion. The primary benefit that BitTorrent provides over a standard client-server model of sharing files is that many data chunks can be downloaded at once, potentially providing better download speeds if the data chunks are properly distributed among peers.

Software is the common method of joining and interacting with the dP2P overlay network, as it manages the technical requirements of participating within a dP2P network, and allows the user to focus on the purpose of the network. Within BitTorrent, this means that the user doesn't need to know which peers have the data required, how to request that data from the necessary peers, or how to re-assemble the data chunks properly.

2.3 Why Distributed Peer-to-Peer Overlay Networks?

Distributed, peer-to-peer overlay networks have one primary benefit over centralized, client-server model overlay networks: they are not dependent on a specific piece of server infrastructure being in place. If the server providing the requested data is unavailable, the user becomes unable to obtain that data until a server hosting the data becomes available. In a distributed, peer-to-peer overlay network, peers may join and leave as they wish, and the network will scale with them. There is no requirement for more infrastructure if a sudden influx of peers wish to join, and there is no wasted infrastructure if the number of peers suddenly drops.

There is, however, a reason that the majority of the Internet fits the traditional client-server model of networking: simplicity. Running a server, and allowing users to choose to interact with it or not; is a much simpler concept than coordinating all of the peers on a network, gracefully managing large changes in the numbers of peers on the network, and structuring the network in such a way that communication between peers is efficient. In the context of file sharing, a traditional client-server model would need to act as a central location responding to data requests, a common communication point, and a single point of interaction in order to request data. This model is much simpler than having no central data location (which then leads to issues of

data consistency/concurrency), choosing which of the peers to download the data from (when every peer on the network is a potential source), and knowing enough about the structure of the network to request this data efficiently (finding the shortest path from peer A to peer B).

These three differences motivate some of the primary research topics within dP2P overlay networks, and will be referred to as the "networking problems" for overlay networks. While these are areas of ongoing reserach, these three attributes of dP2P overlay networks also provide areas of contention within the research community, and provide opportunities for differentiation between types of dP2P overlay networks. The most common differences between dP2P overlay networks focus on issues of data consistency or network architecture. Data consistency, concurrency, and maintaining the proper state among peers is a problem that is inherent in distributed systems as a whole, whereas network structure and architecture is more specific to P2P overlay networks in general. As such, the research community seems to have started differentiating between types of dP2P overlay networks based primarily on the mechanism for how peers choose which other peers to interact with, and how those interactions are optimized for efficiency.

There are also a number of research questions in dP2P overlay networks that address security issues like authentication, encryption, and security among the peers on the network. Given the decentralized nature of dP2P overlay networks, tasks such as authenticating the identity of a user and using secure encryption techniques between peers become more difficult. The standard client-server model of authentication assumes a centralized understanding of users (and their respective access rights/settings within the network), but in removing this centralized aspect of the network, the problem of requesting, verifying, and managing the identity of a user becomes much more difficult. One way of dealing with this issue is discussed in [17], wherein there is a distributed set of authentication servers, and one delegate. When a client needs to authenticate, the client interacts with the delegate, which then interacts within the quorum of servers that are necessary to authenticate the client.

Similarly, the task of encrypting a connection between peers becomes much more difficult when there is no central authority. This task entails authenticating all participating peers to one another, defining a common encryption scheme, and performing the encryption in a way that is secure. Within more traditional networking architectures, a central authority that understands which peers can participate responds to requests for authentication (this is done in a pair-wise way, both the client and the server authenticate to one another, in order to verify the "identity" of both the client and the server) and defines the encryption scheme. In contrast, all peers in a dP2P network need to agree upon a specific encryption technique to use. These will be referred to as the "security problems" within overlay networks. Much of the current research the area of dP2P overlay networks is done on this set of security problems.

3. TYPES OF OVERLAY NETWOKS

There are three primary types of dP2P overlay networks: structured dP2P overlay networks, unstructured dP2P overlay networks, and hybrid dP2P overlay networks. Structured dP2P overlay networks are characterized by the fact that the network as a whole has a mechanism by which the network is structured. These will have a very specific architecture, which is defined by the system. Conversely, unstructured dP2P overlay networks do not have such a mechanism, and instead take a more bottom-up approach, modifying the architecture of the network as required by the scale of the network. These types of overlay networks have no pre-defined structure. In the area between these two extremes, there is a third type of network, the hybrid dP2P overlay network. The hybrid approach combines both approaches, and may have some over-arching structure, but also has the capacity to shift the structure should the situation require it. Hybrid dP2P overlay networks are also referred to as centralized, peer-to-peer networks, as the most common usage hybrid networks have a central server that aids in structuring the network [4].

All dP2P overlay networks are a group of nodes, and these nodes can take on the role of distributor, router, or destination node. The distributor node is a node or machine that is hosting some object, a router is a node that passes along network traffic as it reaches the node, and the destination node is the ultimate endpoint for the network traffic. Nodes may take on one or more of these roles. That is, a node can be both a router and a distributor if they data they are hosting is not the data requested, but if the node is still participating in routing data to other nodes in the network.

Additionally, all dP2P overlay networks must define specific behavior for how a node may join the network, how the network will behave when a node unexpectedly leaves a network, and what needs to occur if traffic does not successfully reach the destination node. In the following sections, we will discuss the specifics of how each type of dP2P network handles these three behaviors.

4. UNSTRUCTURED DISTRIBUTED PEER-TO-PEER OVERLAY NETWORKS

The first type of dP2P overlay networks we will discuss are the unstructured dP2P overlay networks. These do not have an overarching scheme for how peers fine one another, nor for how data is routed between peers. Instead, when a peer wishes to interact with another peer, they must first find a peer to interact with. One well known example of an unstructured dP2P overlay network was the original Kazaa filesharing network. If a peer within the Kazaa network wanted a file, the peer first had to find a second peer that was hosting the file. The peer could then contact the second peer, and begin the download of the requested data. When scaled, popular files became very accessible because they were being distributed so widely throughout the network. However, items that were less popular either were unable to be found, or became very difficult to come across. For this reason, as unstructured dP2P overlay networks begin to get large, their usefulness decreases. In some cases, a peer searching for something would be unsuccessful, because the set of peers hosting popular items significantly overshadowed the set of peers hosting less popular items.

This type of dP2P overlay network, because of it's unstructured nature, cannot guarantee that traffic will reach it's destination. Nor can unstructured dP2P networks guarantee that the path the traffic takes will be optimal. Further, unstructured dP2P overlay networks can incur high bandwidth requirements, because of their need to broadcast data [8].

Further, due to the lack of structure, there is no need to define special join or leave behavior, as nodes can join and leave as they wish without affecting the behavior of the rest of the network. While this allows levels of flexibility between on the network, the downsides of unstructured dP2P overlay networks make other options more desirable.

5. HYBRID DISTRIBUTED PEER-TO-PEER OVERLAY NETWORKS

As an attempt to deal with this issue, the concept of a hybrid dP2P overlay network came into fruition. Given that unstructured dP2P overlay networks may cause searches to fail in some cases, a simple solution is to provide a centralized mechanism to help a peer searching for something. Within the Kazaa example, a centralized lookup index was implemented to allow peers to first ask a server which other peers may be hosting the requested data. The server would then respond with a list of peers known to have hosted this data at one point. Upon receiving this list, a peer would then request the data from peers on the list, allowing the peer to successfully retrieve the data requested, assuming the state of the hosting peers had not changed since the central index was updated.

This method of adding a degree of centralization to an unstructured dP2P overlay network does come with potential downsides, given that, as previously discussed, centralization means a central point of failure. The benefit, however, is that the centralization in this method does not make communication between peers impossible if the central server were to be unavailable. The central server provides a way to augment the capabilities of an unstructured dP2P overlay network, without requiring that the network be dependent on the server.

Due to the lack of overall structure, there is no need to define special join or leave behavior, as nodes can join and leave as they wish without affecting the rest of the network.

There are slightly more complex hybrid architectures, that attempt to deal with the point of failure problem, while also spreading the load across multiple centralized servers. [15] The two server architectures that are most similar to the two extremes (unstructured dP2P overlay networks in Section 4, and structured dP2P overlay networks as described in Section 6) are referred to as the "full replication architecture" and the "hash architecture".

5.1 Full Replication Architecture

The full replication architecture replicates the data that would, in the simple single-server case, be stored on the server to all of the servers being used for the hybrid network. This architecture allows two primary benefits. First, when a request is made, the server that the client interacts with will be able to respond to all of the requests the client is making. This allows the client to only interact with one server, and helps cut down on network overhead. Second, because each server maintains a complete copy of the lookup index, if any one server goes down, others may be available to service requests. [15]

However, this structure does mean that the "concept of the world" that a single server would be able to maintain needs to be replicated across all servers within the hybrid network. One area where this may be an issue is tracking client connections. Consider two servers, Server A and Server B, both are a part of the full replication architecture in a hybrid network. Both Server A and Server B host the same data, allowing users to interact with one or the other, but not necessarily both. If Server A has a sudden influx of users attempting to connect, the user connection statistics must be replicated to Server B. This can lead to significant amounts of network overhead between Server A and Server B, and can Server B to need to process significant amounts of incoming data in order to maintain a common state with Server A. [15]

5.2 Hash Architechture

The hash architecture is similar to the full replication architecture (see Section 5.1) but instead of each server maintaining a fully copy of the lookup index, there are instead indicators within the lookup index that map to the server hosting the required information. A client may make a request, which gets routed to Server A. Upon receiving the request, Server A will check it's lookup index, and may find that the requested data is hosted on a different server (Server B). Server A will then request the information from Server B, collect all the required information, and return all of this to the client. [15]

This architecture also handles the problem of maintaining state across servers (mentioned previously in Section 5.1) in a tiered way. That is, as with looking up data, only a subset of the servers need to be interacted with in order to notify amounts of users requesting data. As with the full replication architecture, this can cause a fair amount of bandwidth usage between servers, but less so than the full replication architecture. There are also ways of making this exchange more efficient. [15]

6. STRUCTURED DISTRIBUTED PEER-TO-PEER OVERLAY NETWORKS

In order for the nodes to minimize overhead and properly route traffic, structured dP2P overlay networks define a structure or algorithm for how traffic is routed throughout the network. Within a traditional network, the routing is done by specialized hardware, and the client is told where to direct the outgoing traffic by a specific network service known as DHCP, which is run by the ISP (Internet Service Provider) providing the client with a connection to the network.

Within this section we will discuss 3 primary structured dP2P overlay networks, each using a slightly different algorithm for maintaining structure within the network, adding and removing nodes, and dealing with failed attempts to route traffic to the destination node. All 3 of these use a variation of a distributed key-value hash lookup, known as a Distributed Hash Table (DHT) [12]. A simple example of a key-value lookup is a dictionary, where the word being looked up is the key, and the value is the definition. Structured dP2P overlay networks use a distributed key-value lookup to obtain information on other nodes, and the key is often a hashed "name" for the node. DHTs are used to help define the structure of the overlay network, and the individual DHT structures each require specific join/leave behaviors.

6.1 Pastry

Pastry [8] is a structured dP2P overlay network developed by Microsoft Research in Cambridge, UK, that was intended as a backend for PAST (a distributed peer-to-peer file-store) [1]. Pastry randomly assigns each node within the network a 128-bit `nodeId`, the key in the DHT, while the value is the return response (e.g. URL to a file being hosted, specific data requested or an XML response) when a request is received by a given node node.

Each node in the Pastry network has what is referred to as a state table. This state table is comprised of three sections, the leaf set, the neighborhood set, and the routing table. The leaf set and the neighborhood set may contain similar elements. The leaf set is a binary search tree containing the L nodes with most similar prefixes and is used for routing. In contrast the neighborhood set is unordered, is not used

for routing, and instead maintains a list of M nodes that are nearest to itself (based on physical network proximity).

The leaf set will tend to have unique `nodeId`s, but the scheme for referring to neighbor nodes is defined by the system being built on top of Pastry. Within PAST, this is the SHA-1 [13] hashing algorithm. This allows the original node to have a diverse set of neighbor node options when deciding how to route the data received. When a node needs to send data across the network, it requires two pieces of information to do so: first, it needs a numeric key (the `nodeId` of the destination node) to be able to make a decision about how to route the information. Second, the node needs to know what information to pass along, referred to as the message. When a node in the Pastry network receives data, it will pass the key and the message along to the neighbor with the numerically closest `nodeId` to the key.

When a new node A attempts to join the Pastry network, it is assumed that the new node will already have knowledge of a nearby node B (one that is already a part of the Pastry network). Node A will then request that node B begin routing a special "join" message, as well as the key for the node A. Node B will then route the message to the node C within its leaf set that has has the `nodeId` most similar to node A. Node C will continue this process until the "join" message reaches the node (Z) in the network that has the most similar `nodeId` to node A. The destination node Z and all nodes along the path to Z will then send their routing tables to node A, which is attempting to join the network. This gives node A a base routing table to work with, which can then be modified as needed. Node A copies node B's neighbor list (and then derives the leaf set from there), as node B is assumed to be physically near to node A.

Given the peer-to-peer nature of the system, it is common for nodes to leave the network without warning. When this occurs, it will not be noticed until node N makes an attempt to route a message through node Z, which has already left. When node N attempts to route information through node Z, and node Z does not respond, then node N will contact the next node (node M) in the row containing Z in N's routing table and request the `nodeId` for the node that corresponds to node Z (in node N's routing table). If node M is also unavailable, node N will traverse the rest of the row, and attempt the same process. In the event that none of the nodes in the row are available, node N will move on to the next row until the process can be completed. While it's possible for this process to fail to find an active node, this is statistically unlikely.

The neighbor list also needs to be updated when a node leaves the network. This is done by polling all nodes listed in a given node's (node A) neighbor list at a pre-determined time interval. If a node does not respond, it will be removed from node A's neighbor list, and node A will request the neighbor lists from all other nodes in it's neighbor list, calculate the approximate distance of the new potential neighbors and add the nearest neighbors until its neighborhood set is full.

6.2 Tapestry

In a similar way to Pastry, Tapestry [16] is also intended to be a lower-level system that other systems are built on top of. Tapestry draws it's design from a method introduced by Plaxton et al., [7] (referred to as the Plaxton method here). Within Tapestry, each node is assigned a 160-bit `nodeId`, and similarly to Pastry, maintains a list of the nearest nodes to it (also referred to as neighbor nodes). However, in contrast to Pastry, Tapestry does not maintain a leaf set or neighborhood set, but instead maintains a list of backpointers (the set of nodes for which the current node is listed as a neighbor). Tapestry also assumes that the set of nodes within the network are evenly distributed, and recommends using the SHA-1 [13] hashing algorithm when assigning `nodeId`s to ensure this.

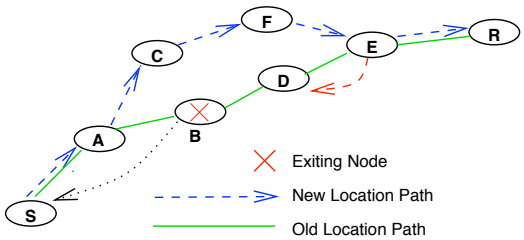

Figure 1: When node *B* attempts to leave the network, it notifies destination node *S*, which steps back through the network to the last known location for a given piece of information, node *R*, and updates the neighbor lists accordingly. Taken from [16].

Similarly to Pastry, Tapestry also routes data through it's peers, however, where Pastry does routing based on numeric proximity to the target key, Tapestry routes based on which of the neighbor nodes has a prefix that most closely matches that of the `nodeId` of the destination node.

Tapestry and Pastry use similar techniques for adding or removing nodes from the network. When a node attempts to join the Tapestry network, it also finds a node on the Tapestry network, which the authors refer to as the gateway node. Unlike in Pastry, however, proximity has no bearing on which node is chosen to be the gateway node in Tapestry.

When a new node N attempts to join the network, it will already have a `nodeId`. It will then attempt to route a message to it's `nodeId` through the gateway node, which allows node N to gain some insight into where it ought to be logically located on the network. Node N will then request the neighbor map from the last node (node L) along the route to N's `nodeId`. This allows node N to have an initial neighbor map. Node N then compares the distance between itself it's neighbors. Node N chooses the neighbor that is closest, and modifies it's neighbor map to reflect these changes. This process continues until there are no better neighbor nodes available than the ones node N is already aware of.

After this process, the backpointers from node L are traversed, and those nodes are notified that node N has been inserted. These nodes then update their own neighbor maps, fully integrating node N into the network.

Removing a node from the Tapestry network is much less involved. Nodes periodically send messages to all of their neighbor nodes, and expect a response back (in order to ensure that all of a given node's neighbors are still available). When a node leaves, it will either be noticed through this process, or it can notify all of it's neighbor nodes using it's backpointers (as demonstrated in Figure 1).

6.3 Chord

While Pastry and Tapestry both structure themselves using a series of interconnected nodes (a series of connected sub-graphs), Chord[9] instead structures its network as a ring. Where Pastry and Tapestry nodes were given an `nodeId`, Chord gives it's nodes an identifier, which is derived by hashing the IP address of the machine. The nodes are then ar-

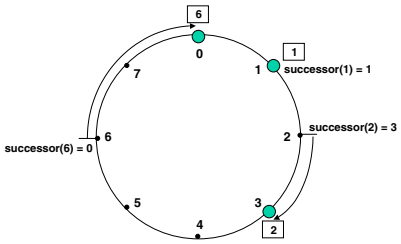

Figure 2: An identifier circle being shown for three individual nodes, where the *successor(X)* function returns the successor for node *X*. Taken from [9].

ranged numerically in what the authors refer to as an identifier circle, by using the node identifier modulo 2^n where n is the number of nodes in the network at that point in time. Then, a key is assigned to the node where the identifier either equals or is the next subsequent identifier of the key. This can be seen in in Figure 2.

This structure for node organization means that nodes can join or leave the network with minimal disruption, as a node only has an understanding of itself, and it's successor. If node A attempts to join the network, keys previously assigned to the node that will become node A's successor get assigned to node A. Likewise, if node A leaves the network, keys that were assigned to node A get assigned to node A's successor.

7. CURRENT RESEARCH

Unfortunately, the "security problems" mentioned in Section 2.3 mean that while dP2P networks currently have the capacity to operate in the same was a a traditional network would, traditional networks have the advantage of a centralized security mechanism. There is a significant amount of research being done into authentication schemes [2, 17], how peers know whether or not they can trust the information they are receiving [10, 14, 5], and other security and trust issues within a distributed system.

8. CONCLUSION

In this paper, we have discussed three major types of distributed peer-to-peer overlay networks in detail. We have described major differences between the three types of dP2P overlay networks, and we have described specific systems within each category. Much of the research being done in dP2P overlay networks is happening in structured dP2P systems (see Section 6), and we believe these systems are more practical than either unstructured dP2P systems (see Section 4) or hybrid P2P systems (see Section 5). The unstructured category of dP2P does not guarantee that data will be accessible, and the hybrid dP2P systems have individual points of failure, which seems to go against the overall goal of a P2P system.

9. REFERENCES

[1] P. Druschel and A. Rowstron. PAST: a large-scale, persistent peer-to-peer storage utility. In *Proc. HotOS VIII*, pages 75–80, 2001.

[2] W. K. Josephson, E. G. Sirer, and F. B. Schneider. Peer-to-peer authentication with a distributed single sign-on service. *Peer-to-Peer Systems III*, pages 250–258, 2005.

[3] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, C. Wells, et al. Oceanstore: An architecture for global-scale persistent storage. *ACM SIGARCH Computer Architecture News*, 28(5):190–201, 2000.

[4] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *ICS '02: Proceedings of the 16th international conference on Supercomputing*, pages 84–95, New York, NY, USA, 2002. ACM.

[5] S. Marti and H. Garcia-Molina. Taxonomy of trust: Categorizing P2P reputation systems. *Computer Networks*, 50(4):472–484, 2006.

[6] G. P. Millar, T. A. Ramrekha, and C. Politis. A peer-to-peer overlay approach for emergency mobile ad hoc network based multimedia communications. In *Proceedings of the 5th International ICST Mobile Multimedia Communications Conference*, pages 1–5, 2009.

[7] C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing nearby copies of replicated objects in a distributed environment. *Theory of Computing Systems*, 32(3):241–280, 1999.

[8] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware 2001: IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg, Germany, November 12-16, 2001. Proceedings*, page 329, 2001.

[9] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, 2001.

[10] K. Watanabe, Y. Nakajima, T. Enokido, and M. Takizawa. Ranking factors in peer-to-peer overlay networks. *ACM Trans. Auton. Adapt. Syst.*, 2, September 2007.

[11] Wikipedia. *BitTorrent (protocol) - Wikipedia, The Free Encyclopedia*. 2010. [Online; accessed 28-October-2010].

[12] Wikipedia. Distributed hash table — wikipedia, the free encyclopedia, 2010. [Online; accessed 28-October-2010].

[13] Wikipedia. *SHA-1 - Wikipedia, The Free Encyclopedia*. 2010. [Online; accessed 28-October-2010].

[14] L. Xiong and L. Liu. A reputation-based trust model for peer-to-peer e-commerce communities. In *E-Commerce, 2003. CEC 2003. IEEE International Conference on*, pages 275–284. IEEE, 2003.

[15] B. Yang, H. Garcia-Molina, et al. Comparing hybrid peer-to-peer systems. In *Proceedings of the International Conference on Very Large Data Bases*, pages 561–570, 2001.

[16] B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. *Computer*, 74:11–20, 2001.

[17] L. Zhou, F. B. Schneider, and R. Van Renesse. Coca: A secure distributed online certification authority. *ACM Trans. Comput. Syst.*, 20:329–368, November 2002.